Pointers: Introduction, Pointers to pointers, Compatibility, L value and R value, Tips and Common Programming Errors, Key Terms, Summary, Practice Set.
Pointer Applications: Arrays, and Pointers, Pointer Arithmetic and Arrays, Memory Allocation Function, Array of Pointers, Programming Application, Tips and Common Programming Errors, Key Terms, Summary, Practice Set.
Processor Commands: Processor Commands, Tips and Common Programming Errors, Key Terms, Summary, Practice Set.

➢ Introduction:-

- One of the powerful features of C is ability to access the memory variables by their memory address.
- This can be done by using Pointers. The real power of C lies in the proper use of Pointers.
- A pointer is a variable that can store an address of a variable (i.e., 112300).We say that a pointer points to a variable that is stored at that address.
- A pointer itself usually occupies 4 bytes of memory (then it can address cells from 0 to 232-1).

Advantages of Pointers:-

1. A pointer enables us to access a variable that is defined out side the function.
2. Pointers are more efficient in handling the data tables.
3. Pointers reduce the length and complexity of a program.
4. They increase the execution speed.

Definition :-

A variable that holds a physical memory address is called a pointer variable or Pointer

Declaration :

Datatype * Variable-name;

```
Eg:-    int *ad; /* pointer to int */
        char *s; /* pointer to char */
        float *fp; /* pointer to float */
        char **s; /* pointer to variable that is a pointer to char */
```

- A pointer is a variable that contains an address which is a location of another variable in memory.

Consider the Statement

p=&i;

Here „&" is called address of a variable.
'p' contains the address of a variable i.

The operator & returns the memory address of variable on which it is operated, this is called Referencing.

The * operator is called an indirection operator or dereferencing operator which is used to display the contents of the Pointer Variable.

Consider the following Statements :

```
int *p,x;
x =5;
p= &x;
```

Assume that x is stored at the memory address 2000. Then the output for the following printf statements is :

	Output
Printf("The Value of x is %d",x);	5
Printf("The Address of x is %u",&x);	2000
Printf("The Address of x is %u",p);	2000
Printf("The Value of x is %d",*p);	5
Printf("The Value of x is %d",*(&x));	5

➢ Pointers to pointers:-

- A pointer to a pointer is a form of multiple indirection, or a chain of pointers. Normally, a pointer contains the address of a variable.
- When we define a pointer to a pointer, the first pointer contains the address of the second pointer, which points to the location that contains the actual value as shown below.

- A variable that is a pointer to a pointer must be declared as such.

- This is done by placing an additional asterisk in front of its name. For example, the following declaration declares a pointer to a pointer of type int –

```
int **var;
```

➢ **Compatibility pointer:-**

The rules for assigning one pointer to another are tighter than the rules for numeric types.

For example, you can assign an int value to a double variable without using a type conversion, but you can't do the same for pointers to these two types. Let's see a simple C program to exemplify this.

```
*
 * ptr_compatibility.c -- program illustrates concept of pointer
 * compatibility
 */
#include <stdio.h>

int main(void)
{
    int n = 5;
    long double x;

    int *pi = &n;
    long double *pld = &x;

    x = n;     /* implicit type conversion */
    pld = pi;  /* compile-time error: assigning pointer-to-int to */
               /* pointer-to-long-double */

    return 0;
}
```

➢ **L value and R value:-**

L-value: "l-value" refers to memory location which identifies an object. l-value may appear as either left hand or right hand side of an assignment operator(=). l-value often represents as identifier.

Expressions referring to modifiable locations are called "**modifiable l-values**". A modifiable l-value cannot have an array type, an incomplete type, or a type with the **const** attribute

In C, the concept was renamed as **"locator value"**, and referred to expressions that locate (designate) objects.

The l-value is one of the following:

1. he name of the variable of any type i.e, an identifier of integral, floating, pointer, structure, or union type.
2. A subscript ([ ]) expression that does not evaluate to an array.
3. A unary-indirection (*) expression that does not refer to an array
4. An l-value expression in parentheses.
5. A **const** object (a nonmodifiable l-value).
6. The result of indirection through a pointer, provided that it isn't a function pointer.
7. The result of member access through pointer(-> or .)

R-value: r-value" refers to data value that is stored at some address in memory. A r-value is an expression that can't have a value assigned to it which means r-value can appear on right but not on left hand side of an assignment operator(=).

Note: The unary & (address-of) operator requires an lvalue as its operand. That is, &n is a valid expression only if n is an lvalue.

- **Arrays:-**
  - An array is defined as the collection of similar type of data items stored at contiguous memory locations.
  - Arrays are the derived data type in C programming language which can store the primitive type of data such as int, char, double, float, etc.
  - It also has the capability to store the collection of derived data types, such as pointers, structure, etc.
  - The array is the simplest data structure where each data element can be randomly accessed by using its index number.
- **Pointer Arithmetic in C:-**

Pointer Arithmetic in C

- We can perform arithmetic operations on the pointers like addition, subtraction, etc.
- However, as we know that pointer contains the address, the result of an arithmetic operation performed on the

pointer will also be a pointer if the other operand is of type integer.

- In pointer-from-pointer subtraction, the result will be an integer value.
- Following arithmetic operations are possible on the pointer in C language:

  - Increment
  - Decrement
  - Addition
  - Subtraction
  - Comparison

Incrementing Pointer in C:-

- If we increment a pointer by 1, the pointer will start pointing to the immediate next location.
- This is somewhat different from the general arithmetic since the value of the pointer will get increased by the size of the data type to which the pointer is pointing.

The Rule to increment the pointer is given below:

new_address= current_address + i * size_of(data type)

Where i is the number by which the pointer get increased.

Decrementing Pointer in C

- Like increment, we can decrement a pointer variable. If we decrement a pointer, it will start pointing to the previous location.
- The formula of decrementing the pointer is given below:

new_address= current_address - i * size_of(data type)

C Pointer Addition

We can add a value to the pointer variable. The formula of adding value to pointer is given below:

new_address= current_address + (number * size_of(data type))

C Pointer Subtraction

Like pointer addition, we can subtract a value from the pointer variable. Subtracting any number from a pointer will give an address. The formula of subtracting value from the pointer variable is given below:

new_address= current_address - (number * size_of(data type))

➢ Memory Allocation Function:-

- The concept of **dynamic memory allocation in c language** *enables the C programmer to allocate memory at runtime*.
- Dynamic memory allocation in c language is possible by 4 functions of stdlib.h header file.

1. malloc()
2. calloc()
3. realloc()
4. free()

Before learning above functions, let's understand the difference between static memory allocation and dynamic memory allocation.

static memory allocation	dynamic memory allocation
memory is allocated at compile time.	memory is allocated at run time.
memory can't be increased while executing program.	memory can be increased while executing program.
used in array.	used in linked list.

Now let's have a quick look at the methods used for dynamic memory allocation.

malloc()	allocates single block of requested memory.
calloc()	allocates multiple block of requested memory.
realloc()	reallocates the memory occupied by malloc() or calloc() functions.
free()	frees the dynamically allocated memory.

➢ Array of Pointers:-

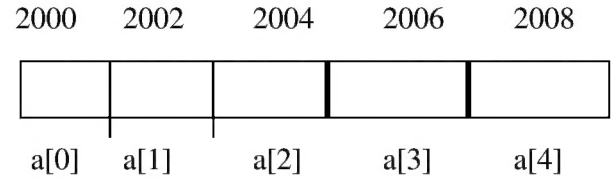
In computer programming, an array of pointers is an indexed set of variables in which the variables are pointers (a reference to a location in memory).

- Pointers are an important tool in computer science for creating, using, and destroying all types of data structures.
- An array of pointers is useful for the same reason that all arrays are useful: it allows you to numerically index a large set of variables.
- Below is an array of pointers in C that sets each pointer in one array to point to an integer in another and then print the values of the integers by dereferencing the pointers.

POINTERS WITH ARRAYS:-

When an array is declared, elements of array are stored in contiguous locations. The address of the first element of an array is called its base address.

Consider the array



The name of the array is called its base address.

i.e., a and k& a[20] are equal

Now both a and a[0] points to location 2000. If we declare p as an integer pointer, then we can make the pointer P to point to the array a by following assignment.

P = a;

We can access every value of array a by moving P from one element to another.

i.e., P points to 0th element
P+1 points to 1st element
P+2 points to 2nd element
P+3 points to 3rd element
P +4 points to 4th element

➢ **Programming Application:-**

APPLICATIONS OF C LANGUAGE

1. C language is used for **creating computer applications**
2. Used in writing Embedded software
3. Firmware for various electronics, industrial and communications products which use micro-controllers.
4. It is also used in developing verification software, test code, simulators etc. for various applications and hardware products.
5. For **Creating Compiler** of different Languages which can take input from other language and convert it into lower level machine dependent language.
6. C is used to implement different Operating System Operations. UNIX kernel is completely developed in C Language.

➢ Processor Commands:-

- The **C Preprocessor** is not a part of the compiler, but is a separate step in the compilation process.
- In simple terms, a C Preprocessor is just a text substitution tool and it instructs the compiler to do required pre-processing before the actual compilation.
- We'll refer to the C Preprocessor as CPP.
- All preprocessor commands begin with a hash symbol (#).
- It must be the first nonblank character, and for readability, a preprocessor directive should begin in the first column.
- The following section lists down all the important preprocessor directives –

Sr.No.	Directive & Description
1	#define Substitutes a preprocessor macro.
2	#include Inserts a particular header from another file.
3	#undef Undefines a preprocessor macro.
4	#ifdef Returns true if this macro is defined.
5	#ifndef Returns true if this macro is not defined.
6	#if Tests if a compile time condition is true.
7	#else The alternative for #if.
8	#elif #else and #if in one statement.
9	#endif Ends preprocessor conditional.
10	#error Prints error message on stderr.
11	#pragma Issues special commands to the compiler, using a standardized method.